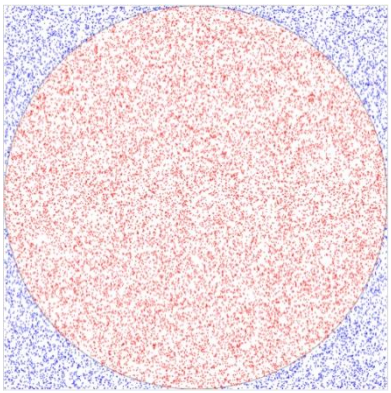
Lab Report 2

CSE455 - High Performance Computing

REDA MOHSEN REDA
18P5141
CESS

Problem Statement

We need to calculate PI. One method to calculate PI (3.141592…) is by using Monte Carlo method.



Introduction About Solution

- we have a circle of radius 0.5, enclosed by a 1 × 1 square.
- The area of the circle is $\pi r^2 = \pi/4$, The area of the square is 1. if we divide the area of the circle by the area of square, we get $\pi/4$.
- We then generate a large number of uniformly distributed random points and count the number of points that fall inside the circle.
- When we divide the total number of points and the number of points inside circle, we get a value that is an approximation of the ratio of the areas we calculated $\pi/4$.

$$\frac{\pi}{4} \approx \frac{N_{inner}}{N_{total}}$$

$$\pi \approx 4\frac{N_{inner}}{N_{total}}$$

Solution

Code

```c
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


int main(int argc, char* argv[])
{

    int nPoints = 10000000;
    int tid, nthreads = 8;
    double nInside = 0;
    double rand_x, rand_y;

#pragma omp parallel private(tid) reduction(+: nInside) num_threads(nthreads)
    {
        //get the thread id number
        tid = omp_get_thread_num();

        int start = tid * (nPoints / nthreads);
        int end = (tid + 1) * (nPoints / nthreads);

        for (int i = start; i < end; i++) {
            //random value for x and y
            rand_x = (double)(rand() % 100) / 100.0;
            rand_y = (double)(rand() % 100) / 100.0;

            //check if the point inside the circle or not
            if (sqrt(pow(rand_x, (double)2) + pow(rand_y, 2)) <= 1) {
            //if inside the circle, increment the Inside variable
                nInside++;
            }
        }
    }
    // print PI value which equals (#points inside circle / total #points) * 4
    printf("PI equals %f", (nInside / nPoints) * 4);
}
```

```
PI equals 3.180747
```